
	<p style="text-align: center;">MINISTÉRIO DA EDUCAÇÃO UNIVERSIDADE FEDERAL DO PIAUÍ - UFPI CAMPUS SENADOR HELVÍDIO NUNES DE BARROS - CSHNB CURSO DE SISTEMAS DE INFORMAÇÃO Disciplina: Estrutura de Dados I Professora: Luana Batista da Cruz</p>	
---	--	---

SEGUNDA AVALIAÇÃO (EM DUPLA) – VALE 7 PONTOS

1. Faça um programa em C que tenha um menu de opções para cadastrar, buscar e imprimir cursos e suas disciplinas. Para isto, crie um vetor de curso, onde cada curso deve ter: código do curso, nome, quantidade de períodos e carga horária total, turno (manhã, tarde, noite, integral). Além disso, um vetor de disciplinas contendo: código da disciplina, código do Curso, nome, período a qual ela pertence, carga horária da disciplina (valor múltiplo de 15). Os vetores devem ser ordenados pelo seus códigos. As opções do menu devem ser **(1,5 pontos)**:
 - a) Cadastrar Curso: faça uma função para cadastrar cursos, não permitir código de curso repetido. Lembre-se, que pode-se ter 2 cursos com o mesmo nome, mas não com o mesmo código. Por exemplo: pode ser 2 cursos de Pedagogia um matutino e outro noturno. Manter sempre o vetor ordenado pelo código do curso.
 - b) Cadastrar Disciplina: faça uma função para cadastrar disciplinas, uma disciplina só pode ser cadastrada para um curso que já existe, ou seja, deve ser verificado se o curso cadastrado para a disciplina já foi cadastrado no vetor de curso. O período de uma disciplina não pode ser um período inexistente no curso cadastrado. Por exemplo: o curso de Sistemas de Informação tem 8 períodos, logo uma disciplina não pode ser cadastrada no nono período. Manter sempre o vetor ordenado pelo código da disciplina.
 - c) Mostrar um curso: faça uma função que dado o código de um curso devolva os dados do curso e então mostre na tela. Caso não exista o código cadastrado emita uma mensagem para o usuário.
 - d) Mostrar uma Disciplina: faça uma função que dado o código de uma disciplina devolva os dados da disciplina e então mostre na tela. Caso não exista o código cadastrado emita uma mensagem para o usuário.
 - e) Mostrar todos os cursos de um turno: faça uma função que imprima todos os cursos de um determinado turno.
 - f) Mostrar todas as disciplinas de um Curso: faça uma função que imprima todas as disciplinas de um curso. Emita uma mensagem caso o curso solicitado não exista.
 - g) Mostrar disciplinas de um período: faça uma função que dado um período e um curso imprima todas as disciplinas daquele período daquele curso. Emita uma mensagem caso o curso ou período solicitados não exista.

- h) Remover uma disciplina: faça uma função que remova uma disciplina de um curso dado o código da disciplina.
 - i) Remover um curso: faça uma função que remova um curso dado o código do curso. Neste caso só é possível remover um curso que não tenha nenhuma disciplina cadastrada para ele. Caso contrário emita uma mensagem.
Obs.: Utilize o método de ordenação Insert e o método de busca binária quando a busca for pelo código do curso ou pelo código da disciplina.
2. Faça um programa em C que leia um conjunto de livros (vetor de ponteiros), onde cada um dos livros possui as seguintes informações: código, título, autor, editora, ano de edição e números de exemplares (1,0 ponto).
- a) O usuário poderá inserir quantos livros desejar desde que não ultrapasse o limite máximo do conjunto.
Obs.: parâmetro deve ser do tipo “ponteiro para ponteiro”
Obs.: não esqueça de inicializar com NULL
 - b) Faça uma opção onde o usuário digite a editora e o programa mostre todos os livros cadastrados daquela editora e repita o mesmo para autor.
 - c) Faça uma opção para remover os dados de um livro informado pelo usuário.
Obs.: libere o espaço de memória utilizado para os dados do livro.
3. Escreva uma função em C que recebe uma string de caracteres alocada dinamicamente e uma letra e devolve um vetor de inteiros contendo as posições (índices no vetor da string onde a letra foi encontrada) e um inteiro contendo o tamanho do vetor criado (total de letras iguais encontradas). Utilize o retorno de um vetor para retornar os índices e um ponteiro para guardar o tamanho do vetor criado (1,0 ponto).
4. Desenvolva um algoritmo que calcule a quantidade de vezes em que cada número presente em uma determinada matriz ocorra. Esta matriz será de números inteiros e suas dimensões serão informadas pelo usuário. Para alimentar a matriz o programador deve usar a função `rand() % 10000`. A função abaixo deve existir e ser usada seguindo o seu escopo definido abaixo (1,0 ponto):
- ```
int *ocorrencias(int **mat, int l, int c){};
```
- A saída no método `main()` deve obedecer a seguinte estratégia:
- Valor 3 ocorrencias 2  
Valor 5 ocorrencia 1  
Valor 34 sem ocorrencia  
...
- Obs.:** Utilize alocação dinâmica.
5. Crie um programa que implemente o jogo “Bingo de ED I”. Nesse jogo, o jogador deve selecionar a quantidade de números que ele gostaria de apostar (entre 1 e

20), e em seguida, informar os números escolhidos (valores entre 0 e 100). Após receber a aposta, o computador sorteia 20 números (entre 0 e 100) e compara os números sorteados com os números apostados, informando ao apostador a quantidade de acertos e os números que ele acertou. O seu programa deverá implementar as funções ler\_aposta, sorteia\_valores e compara\_aposta (1,5 pontos).

- a) **Ler\_aposta:** deve receber como parâmetro a quantidade de números que serão apostados e um vetor previamente alocado dinamicamente para armazenar a quantidade exata de números apostados. A função deve pedir para o usuário digitar os números apostados e armazená-los no vetor, garantindo que somente números dentro do intervalo de 0 a 100 sejam digitados. A função deve seguir o seguinte protótipo:

**void ler\_aposta(int \*aposta, int n)**

- b) **Sorteia\_valores:** deve receber como parâmetro a quantidade de números que serão sorteados e um vetor previamente alocado dinamicamente para armazenar a quantidade exata de números sorteados. A função deve sortear aleatoriamente os números (entre 0 e 100) e armazená-los no vetor. A função deve seguir o seguinte protótipo:

**void sorteia\_valores(int \*sorteio, int n)**

- c) **Compara\_aposta:** deve receber como parâmetro o vetor com os números apostados (aposta), o vetor com os números sorteados (sorteio), juntamente com os seus respectivos tamanhos (na e ns) e um ponteiro para uma variável inteira (qtdacertos), onde deve ser armazenada a quantidade de acertos. A função deve retornar o ponteiro para um vetor alocado dinamicamente contendo os números que o apostador acertou. A função deve seguir o seguinte protótipo:

**int\* compara\_aposta(int \*aposta, int \*sorteio, int \*qtdacertos, int na, int ns)**

**Obs.:** crie a função principal do programa utilizando as funções criadas anteriormente para implementar o jogo “Bingo de ED I”. Lembre-se que os vetores aposta, sorteio e acertos devem ser alocados dinamicamente e a memória alocada deve ser liberada quando ela não for mais utilizada.

6. Implemente um TAD **Jogador de Futebol**. Cada jogador possui os campos nome, jogos, gols e assistências. Implemente as operações (1,0 ponto):

- **Atribui:** atribui valores para os campos.
- **Imprime:** imprime os dados do jogador.
- **Soma:** soma estatísticas de dois jogadores.
- **EhBom:** testa se o jogador é bom (defina seu próprio critério de bom jogador).
- **Libera:** libera a memória alocada para o jogador de futebol.

**Obs1.:** Utilize alocação dinâmica.

**Obs2.:** Crie o main para testar seu TAD.

**Equipe:** os programas podem ser feitos em dupla, mas os relatórios são individuais. A dupla deve ser identificada no envio do código.

**Data de entrega:** [18/11](#).

**Entregar:** código fonte + relatório (comentários da lógica do funcionamento das questões de forma detalhada).

**Entrevista individual:** agendar horário na [planilha disponibilizada](#).